Machine learning for everyone

Gabriel Vigliensoni-Martin

gabriel.vigliensonimartin@mcgill.ca

Distributed Digital Music Archives & Libraries Lab
Schulich School of Music
McGill University
Learning machines for everyone

Gabriel Viglienson-Martin
gabriel.vigliensonimartin@mcgill.ca
Distributed Digital Music Archives & Libraries Lab
Schulich School of Music
McGill University
Machine learning

• What is learning?
  • Memorizing vs learning

• Machine learning
  • Science of getting computers to act without being explicitly programmed
  • Study of algorithms that learn from examples and experiences instead of hard-coded rules

• Machine learning tasks
  • Classification: output is discrete categories
  • Regression: output is a real-value number
How to write code to tell the difference between an orange and an apple?

Take an input image, do some analysis, and outputs the types of fruit.

Writing lots of manual rules: count how many green pixels there are in the image and compare that to the number of red pixels? The ratio should give a hint about the type of fruit.

Works fine for simple images, but the world is messy and rules start to break. What about green apples? Or black and white photos? Or other fruits?

Colors, edges, shapes, guessing textures, …

New problem may imply a new set of rules!

Machine learning

• How to **write code to tell the difference** between (classify) an orange and an apple?
  • Take an **input** image, do some **analysis**, and **output** the types of fruit
  • Writing **lots of manual rules**
    • Count how many **green pixels** there are in the image and compare that to the number of **red pixels**? The **ratio** should give a **hint about the type of fruit**.
    • **Works fine for simple images**, but the world is messy and rules start to break. What about red apples? Or other fruits?
    • Colours, edges, shapes, guessing textures, …
  • New problem may imply **a new set of rules**!
• We need an **algorithm** that can figure out the rules for us!

• For this, we **train a classifier** (a function that takes some data and assign a label to it as output)

• One big distinctions in machine learning: **supervised learning** versus **unsupervised learning**
Machine learning

Supervised learning

• Supervised learning approaches use examples of the problem you want to solve
• These methods create a classifier by finding patterns in these examples
Machine learning
Supervised learning
Supervised learning steps based on features:

- **collect training data** (examples of the problem we want to solve)
- **take some measurements**. In the context of machine learning these measurements are called **features** (e.g., go to an orchard and measure the weight, shape, and texture of fruits, and then label them)
- put all training data into a table, **the more training data the better the classifier** (model)
- **train the classifier with this data**. There are many training methods, but the inputs and outputs are always the same
A classifier can be seen as a box of rules, and so the training algorithm is the procedure to learn those rules.

It does so by finding patterns in the training data.

For example, it may notice that oranges tend to weigh more, and so it creates a rule saying the more heavy the fruit is the more likely it is an orange.
• This procedure is the same for a new problem. **By changing the training data we can create a new classifier.** There is no need of writing new rules!

• Questions
  • How much training data is needed?
  • How is the optimization achieved?
  • What makes a good feature?

• **“all models are wrong, but some are useful.”** (George E. P. Box)

• The goal of ML is never to make “perfect” guesses, because ML deals in domains where there is no such thing

• **The goal of ML is to make guesses that are good enough to be useful**
In **unsupervised learning** there is **no such thing such a class label**

- We provide a set of **training examples** that **we believe contains internal patterns**
- We leave it to the system to **discover those patterns on its own**
Machine learning
Training and testing architecture

• The purpose of creating a model or classifier is not to classify the training set, but to classify the data \textit{whose class we do not know}.

• We want to create models \textit{that are generalizable}.

Wrongness

- The error rate is known as the cost function (a.k.a., loss function or error)
Neuron

Artificial Neuron

\[ y = \sum_i w_i x_i = w^T x \]
Neural Networks

Iterative optimization

• How to **learn the weights** of a neuron?
• By using a **learning algorithm**
  • learning turns to be a **numerical optimization of weight** parameters
  • The actual output values **get closer** to the target values **in each iteration**
  • Many quite **different sets of weights may work** well
Simplest example: linear neuron with error measure

- A linear neuron has a real-valued output that is a weighted sum of its inputs

- The aim of the learning is to minimize the error summed over all training cases

\[ y = \sum_{i} w_i x_i = w^T x \]
Learning as iterative optimization

• **Toy example** to illustrate the iterative method:
  • Each day you get lunch at the cafeteria
    • Diet consists of **protein**, **salad**, and **carbohydrate**
    • You get **several portions** of each
  • The cashier only tells you the **total price of the meal**
  • After a few days, you ought to be able to **figure out what the price is for each portion** of each kind of food based on all previous examples
Let’s suppose that the true weights that the cashier using to figure out the price, are $1.5$ for a portion of protein, $0.5$ for a portion of carbohydrate, and $1.0$ for a portion of salad.

We start with guesses for these prices and then we adjust the guesses slightly, so that we agree better with what the cashier says.

The prices of the portions are like the weights of a linear neuron.

$$w = \left( w_{\text{protein}}, w_{\text{carbo}}, w_{\text{salad}} \right)$$

Learning as iterative optimization
Learning as iterative optimization

True weights used by the cashier

- 2 portions of protein
- 5 portions of carbohydrate
- 3 portions of fruit

Meal price = 8.5$

Linear neuron
Learning as iterative optimization

Starting with guesses

2 portions of protein

5 portions of carbohydrate

3 portions of fruit

linear neuron

0.5

0.5

0.5

Actual price = 8.5$

Computed price = 5.0$

Residual error = 3.5

Delta rule: $\Delta w_i = \varepsilon x_i (t - y)$

$\Delta w_i = 1/35 \times 3.5 = 0.1 x_i$

Weight changes: (0.2, 0.5, 0.3)

New weights: (0.7, 1.0, 0.8)
Learning as iterative optimization

First iteration

2 portions of protein

5 portions of carbohydrate

3 portions of fruit

0.7

1.0

0.8

linear neuron

Actual price = 8.5$
Computed price = 8.8$
Residual error = -0.3

Delta rule: $\Delta w_i = \epsilon x_i (t - y)$
$\Delta w_i = \frac{1}{35} x_i (-0.3) = -0.001 x_i$

Weight changes: (-0.002, -0.005, -0.003)
New weights: (0.698, 0.995, 0.797)
Learning as iterative optimization

Second iteration

- 2 portions of protein
  - 0.698

- 5 portions of carbohydrate
  - 0.995

- 3 portions of fruit
  - 0.797

linear neuron

Meal price = 8.76$
Residual error = -0.26

... repeat until no significant improvement is reached
• But the delta rule is **applied to all training cases**, not just one, and so the weights are learnt from all training cases

\[ \Delta w_i = \sum_n \varepsilon x_i^n (t^n - y^n) \]
Neural Networks

small change in any weight (or bias)
causes a small change in the output

inputs

Input layer

Hidden layer

Output layer

output + Δoutput

http://neuralnetworksanddeeplearning.com/chap1.html
Deep Neural Networks

Deep Neural Networks
Tensorflow online example

• Tensorflow playground
ML benchmark datasets

- UCI ML Repository
- Deep Learning datasets
  - MNIST online (benchmark) (human performance)
  - CIFAR10 online (benchmark)
Thanks!